Sparsity and L1 regularization

There is a danger of over-fitting when fitting a model to high-dimensional feature vectors. One regularization strategy is to ignore some of the features, either by explicitly removing them, or by making any parameter weights connected to these features exactly zero.

"Sparse" solutions, with many parameters set to zero:

- can be more interpretable,
- can require less memory and less computation,
- and *might* generalize better (but also often not!).

If we have D-dimensional feature vectors, we could fit a model based on each possible subset of these features, and evaluate them on a validation set. What would be the computational cost of this approach, and what else is wrong with it for large D?

Another approach could be a 'greedy' strategy: select the single feature that gives the model that gives the best validation score; then add the single feature that improves the validation score of the model the most, and keep going until the validation score no longer improves. How expensive would this greedy strategy be?

...I'm sure you could come up with better ways to do feature selection, and there are a large number of possibilities. This note will only look at one of them.

L1 regularization

A L1 regularized cost function adds a multiple of the sum of the absolute values of the parameters:

$$C(\mathbf{w}) = E(\mathbf{w}) + \lambda \sum_d |w_d| = E(\mathbf{w}) + \lambda \|\mathbf{w}\|_1.$$

where $E(\mathbf{w})$ is the error on the training set, such as square error or negative log-likelihood, as a function of the parameters.

Like L2 regularization, we penalize weights with large magnitudes. However, the solutions are qualitatively different: with L1 regularization some of the parameters will often be exactly zero, which doesn't usually happen with L2 regularization.

When a weight w_d is zero, the derivative of the L2 penalty is zero: a small change has approximately no effect on the regularization term, but can usually make a small improvement to the training error. Thus an L2 regularized weight will only be set to zero if the training error gradient is zero for that weight.

In contrast, for large enough λ, a small change in a zero weight will increase the L1 regularization term by more than the cost function decreases. For large enough λ, all of the weights will be set to zero, and for moderate λ some of the weights will be.

There is also a graphical explanation of why some of the weights are set to zero. In parameter space we can mark the point representing the optimal weights. We then draw the contours of both the training error $E(\mathbf{w})$ and the regularizer $\|\mathbf{w}\|_1$ that pass through this point. These two contours meet at this point, but don't cross. Proof by contradiction: If the contours crossed we would be able to move along one of the contours, improving one of the two terms in the cost function while leaving the other one constant. That would mean that the point we started at wasn't the optimum after all.

Now we know that the solution will be at a point where the contours of the error and regularizer meet, we can sketch these contours. Usually the contours of a training error will touch a contour of the L1 regularizer at a 'corner', where one or more of the parameters are zero (Murphy Figure 13.3). The contours of the L2 regularizer are spherically symmetric, and so there is no particular reason to think that the contours will meet at an axis, where a parameter is zero.

Bayesian predictions do not remove parameters

Regularized maximum likelihood is sometimes interpreted as maximizing the posterior probability of the parameters ("MAP"), where for L1,

$$p(\mathbf{w}) \propto e^{-\|\mathbf{w}\|}.$$

Using this prior, there is zero probability mass on vectors with parameters set to zero. We believe that each parameter is definitely non-zero. The prior and posterior probability mass associated with the set of vectors where the parameters are all non-zero is one.

There are priors that express a belief that some of the parameters might be zero. For example, the 'spike and slab' prior:

$$p(w_d) = \alpha\delta(w_d) + (1-\alpha)\mathcal{N}(w_d; 0, \sigma_w^2),$$

where $\delta(\cdot)$ is a Dirac delta function. This prior has a 'spike' at zero with mass α, and a Gaussian 'slab' with the remaining probability mass.

However, for any prior that has some probability on non-zero parameters, the predictions of a model

$$p(y \mid \mathbf{x}, \mathcal{D}) = \int p(y \mid \mathbf{x}, \mathbf{w})\, p(\mathbf{w} \mid \mathcal{D})\, \mathrm{d}\mathbf{w},$$

will consider non-zero settings of the parameters. As a result, predictions depend on every feature x_d.

As listed at the start of the note, sparse solutions are often motivated by reasons other than prediction performance. While encouraging sparsity *can* be a useful regularizer, insisting on sparsity often gives worse prediction performance.[1]

Why L1 and what next?

The L1 regularizer is popular because it gives sparse solutions and it is convex. If the error function is also convex it is possible to find the global optimum, but not with simple gradient descent (why?). We give one of the many possible optimization algorithms in the next note.

In some cases, the selection of non-zero weights can be shown to be consistent: with enough data, only features that are actually irrelevant will be pruned out. In practice the non-zero weights are often shrunk too much. Alternatives include fitting with an L1 regularizer, and then refitting the selected non-zero weights with an L2 regularizer. There are also regularizers such as the "elastic net", which include L1 and L2 terms in one cost function.

What you should know

You should know what the L1 regularizer is, why it is used, and why Bayesian predictions don't prune out any features. There are also some questions throughout this note that test your ability to reason about models and methods in this course.

1. For example the Google research, Ad Click Prediction: a View from the Trenches, McMahan et al., KDD 2013. This paper shows you have to be careful to find a regularizing strategy that reduces the number of features without harming performance. As another example, Radford Neal (University of Toronto) won a challenge at a 2003 NIPS workshop that was designed to test feature selection methods. However, he used a Bayesian approach, and so didn't do any feature selection!